Comparative Study of Effects of Delay in Load Balancing Scheme for Highly Load Variant Interactive Applications

Prof. Sudarshan Deshmukh[1] and Sampada S Kalmankar[2]

[1] Pimpri Chinchwad College of Engineering, Department of Computer Engineering, Pune, India

[2] Pimpri Chinchwad College of Engineering, Department of Computer Engineering, Pune, India

Email: [1]deshmukh.sudarshan@gmail.com ,[2]sampadakalmankar@yahoo.com

***Abstract*— Distributed computing architectures utilize a set of computational elements (CEs) to achieve performance that is not attainable on a single CE. Conventional load balancers have proven effective in increasing the utilization of CPU, memory, and disk I/O resources in a Distributed environment. However, most of the existing load-balancing schemes ignore network resources, leaving an opportunity to improve the network resources like delay or effective bandwidth of networks running parallel applications.Load balancing becomes more challenging in interactive applications as load variation is very large and the load on each server may change dramatically over time, by the time when a server is to make the load migration decision, the collected load status from other servers may no longer be valid. This will affect the accuracy, and hence the performance, of the load balancing algorithms. All the existing methods neglect the effect of network delay among the servers on the accuracy of the load balancing solutions. In this paper, due to the change in the load of the server, network delay would affect the performance of the load balancing algorithm will be discussed.**

***Index Terms*— Load balancing, Delay, CPU Resources, I/O resources, Computing Elements, Tasks**

I. Introduction

The performance of LB in delay-infested environments depends upon the selection of balancing instants as well as the level of load-exchange allowed between nodes. A number of load-balancing schemes have been developed, primarily considering a variety of resources, including the CPU, memory, disk I/O, or a combination of CPU and memory resources. These approaches have proven effective in increasing the utilization of resources, assuming that network interconnects is not potential bottlenecks [2]. For example, if nodes have inaccurate information about the state of other nodes, due to random communication delays between nodes, then this could result in unnecessary periodic exchange of loads among them. Consequently, certain nodes may become idle while loads are in transit, a condition that would result in prolonging the total completion time of a load.In general, load-balancing techniques fall into two categories.

A. Centralized Load Balancing

In centralized methods, a central server makes the load migration decisions based on the information collected from all local servers and then passes the decisions to local servers for them to carry out the load migrations.

B. Decentralized Load Balancing

In decentralized methods, each local server could make load migration decisions with information collected from its neighbour servers.Decentralized methods are more efficient than centralized methods as they perform the load balancing process by considering only the local load information. For interactive applications the load on each server may change dramatically over time, by the time when a server is to make the load migration decision, the collected load status from other servers may no longer be valid [1].

Two types of load balancing algorithms [10]:

C. Static Load-Balancing

In this method, allocation is made when the process is created and cannot be changed during process execution to make to make changes in the system load.

D. Dynamic Load Balancing

In this the workload is distributed among the nodes at runtime. Unlike static algorithms, dynamic algorithms allocate processes dynamically when one of the processors becomes under loaded.

II. Literature Survey

There are many researches focusing on the issue of distributed load balancing for CPU and memory resources. Harchol-Balter and Downey [5] developed a CPU-based pre-emptive migration policy which shown to be more effective than non pre-emptive migration policies. Zhang et al. [6] studied load-sharing policies which consider both CPU and memory services among the nodes. An I/O-aware load balancing scheme is proposed by Xiao Qin to meet the needs of a cluster system with a variety of workload conditions [7].Above approaches does not consider the balancing of communication load. A communication-sensitive load balancer was proposed by Cruz and Park [8]. Communication aware load balancing scheme attempts to simultaneously balance two different kinds of I/O load, namely, communication and disk I/O [2].Different load balancing techniques which are differentiated based on the centralized approach and on the decentralized approach, as well as those that concern the delays have been proposed. In decentralized

DOI: 03.LSCS.2013.1.37

approach is that local servers, i.e., servers managing a group of nodes, perform the load balancing process individually. Each server will determine the amount of load to be transferred its neighbour servers.

In traditional load balancing application domains that consider the delay in the design of load balancing are adaptive mesh analysis and queuing analysis. In adaptive mesh analysis, [4] proposes a dynamic load balancing scheme. A method is divided into global and local load balancing processes, with the objective of minimizing remote communications by reducing the number of load balancing processes among distant processors.

In queuing analysis, [9] conduct extensive analysis and reveal that computational delays and load-transfer delays can significantly degrade the performance of load balancing algorithms that do not account for any delays. Further extension of this work is proposed to consider the random arrivals of the external tasks. Both of them try to minimize the task completion time due to network and/or computational delays. On the contrary, [1] argue here that when the local servers have received the load balancing solutions from the central server after some network delay, the loads of the local servers may be very different and the load balancing solutions may no longer be accurate.

III. System Study

A. Communication - Aware Load Balancing for Parallel Applications on Clusters:

In this paper, they have focussed on designing an approach at the software level to achieve high effective bandwidth communication without requiring any additional hardware is proposed. COM-aware load balancing enables cluster to utilize most idle, or underutilized, network resources while keeping the usage of other types of resources reasonably high.

An application model is introduced, which aims at capturing the typical characteristics of the communication, disk I/O, and CPU activity within a parallel application. It is applicable for both communication and I/O intensive parallel applications. The execution time model for a parallel job running on a dedicated cluster environment can be derived as follows. Given a parallel job with p identical processes, the execution time of the job can be calculated as

$$T_p = S_p + \sum_{i=1}^{N} \{MAX_{j=1}^{p}(T_{j,CPU}^{i} + T_{j,COM}^{i} + T_{j,Disk}^{i})\} \quad (01)$$

Where $T_{j,CPU}^{i}$, $T_{j,COM}^{i}$, $T_{j,D}^{i}$ denote the execution time of process j in the i^{th} phase on the three prospective resources.

Communication aware load-balancing scheme:

A dynamic, communication aware load-balancing scheme for non dedicated clusters has been proposed. To measure the communication load imposed by these processes. Let a parallel job formed by p processes be represented by $t_0, t_1, \ldots, t_{p-1}$ and where n_i is the node to which t_i is assigned. Assuming that t_0 is a master process, and $t_j (0 < j < p)$ is a slave process. Let $L_{j,p,COM}$ denote the communication load induced by t_j, which can be computed with the following formula, where $T^{i}_{j,COM}$ is the same as the one used in (01):

$$L_{j,p,COM} = \begin{cases} \sum_{i=1}^{N} T_{j,COM}^{i} & j \neq 0, n_j \neq n_0 \\ 0 & j \neq 0, n_j = n_0 \\ \sum_{1<k<p, n_k \neq n_j} \sum_{i=1}^{N} T_{k,COM}^{i} & j = 0 \end{cases} \quad (02)$$

The communication load on node i, $L_{i,COM}$, as follows:

$$L_{i,COM} = \sum_{\forall j: n_j = i} L_{j,p,COM} \quad (03)$$

To balance the communication load the processes need to select remote nodes, the following two criterions must be satisfied to avoid useless migrations:

Let nodes i and j be the home node and candidate remote node for process t.

Criterion 1: is to bes satisfied to guarantee that the load on the home node will be effectively reduced without making other nodes overloaded. This can be formally expressed as:

$$(L_{i,COM} - L_{j,COM}) > L_{t,p,COM} \quad (04)$$

Criterion 2: Estimated response time of t on node j is less than its local execution. If and are the estimated response time of t on nodes i and j, respectively. $R_{t,}^{i,j}$ the migration cost for process t.

$$R_t^j < R_t^i + R_{t,mig}^{i,j} \quad (05)$$

The experimental results show that the COM-aware approach can improve the performance by up to 206 and 235 percent, in terms of slowdown and turn-around time, respectively, under high communication demands.

B. On Delay Adjustment for Dynamic Load Balancing in Distributed Virtual Environments

As DVE systems are highly interactive and the load on each server may change dramatically over time, by the time when a server is to make the load migration decision, the collected load status from other servers may no longer be valid. Due to communication delays among servers, the load balancing process may be using outdated load information from local servers to compute the balancing flows, while the local servers may be using outdated balancing flows to conduct load migration.

A dynamic load balancing method based on the centralized approach is proposed. The load distribution of the *n* servers of the DVE system is therefore $\{l_1, l_2, l_3, \ldots, l_n\}$. To formulate a server graph is constructed as $G = (S;E)$, The server graph is a weighted undirected graph. The weight associated with each node *Si* is represented by its load l_i and the weight associated with each edge $e_{i,j}$ is called *diffusion coefficient* denoted by $c_{i,j}$. The balancing flow can then be formulated as follows:

$$\bar{l} = l_i - \sum_j \lambda_{i \to j} \quad (06)$$

DOI: 03.LSCS.2013.1.37

Where $\bar{l}$ is the average load over all nodes.

The global heat diffusion algorithm has two main processes:

Global Migration Planning (GMP): It is to compute the balancing flows to solve the load balancing problem. This process is carried out by the central server.

Local Load Transfer (LLT): it is to transfer load between adjacent servers based on the amount indicated by the balancing flows. This process is carried out by each local server that manages a partition.

The Delay Adjustment Schemes

1. Uniform Adjustment Scheme

Due to the communication delay, the balancing flow received by S_i is given by:

$$\lambda'_{i\to j}(t) = \lambda_{i\to j}(t-\Delta t_i, t+\Delta t_i) \quad (07)$$

Now, we adjust the balancing flow as follows:

$$\lambda'_{i\to j}(t+\Delta t_i) = \lambda_{i\to j}(t) + \Delta l_i(t-\Delta t_i, t+\Delta t_i) \quad (08)$$

Where $\lambda'_{i\to j}(t+\Delta t_i)$ is the adjusted balancing flow in S_i at $(t+\Delta t_i)$

This is only an approximation method, which assumes that the contribution to the net increase of S_i's load is uniform among S_i's neighbor servers.

2. Adaptive Adjustment Scheme

The load variation of *Si* can be computed as:

$$\begin{aligned}&\Delta l_i(t-\Delta t_i, t+\Delta t_i)\\ &= \beta_{j\to i}(t-\Delta t_i, t+\Delta t_i) - \beta_{i\to j}(t-\Delta t_i, t+\Delta t_i)\\ &+\beta_{k\to i}(t-\Delta t_i, t+\Delta t_i) - \beta_{i\to k}(t-\Delta t_i, t+\Delta t_i)\end{aligned} \quad (09)$$

Where $\beta_{j\to i}(t-\Delta t_i, t+\Delta t_i) - \beta_{i\to j}(t-\Delta t_i, t+\Delta t_i)$, $\beta_{k\to i}(t-\Delta t_i, t+\Delta t_i) - \beta_{i\to k}(t-\Delta t_i, t+\Delta t_i)$ indicate the net increase in load of S_i contributed by S_j and S_k, respectively. Here, an adjustment scheme is introduced that does not require server-to-server communication.

Thus using the above equations an adjusted balancing flow is given as:

$$\begin{aligned}&\Delta\lambda'_{i\to j}(t+\Delta t_i) =\\ &\lambda_{i\to j}(t) + \beta_{j\to i}(t-\Delta t_i, t+\Delta t_i) - \beta_{i\to j}(t-\Delta t_i, t+\Delta t_i)\end{aligned} \quad (10)$$

Experimental results show that both adjustment schemes can greatly improve the performance of the load balancing, with the adaptive adjustment scheme performs even better on average in the experiments.

C. The Effect of Time Delays on the Stability of Load Balancing Algorithms for Parallel Computations

The main objective that has been proposed is to analyse the effects of delays in the exchange of information among computational elements (CE), and the constraints these effects impose on the design of a load balancing strategy. A deterministic dynamic nonlinear time-delay system is developed to model load balancing. The model is shown to be self consistent in that the queue engths cannot go negative[3] and that the total number of tasks in all the queues and the network is conserved (i.e., load balancing can neither create nor lose tasks).

C. Methodology

The computers are assigned an equal number of homogenous tasks. Some of the nodes generate more tasks and very quickly the loads on various nodes become unequal. To balance the loads, each computer in the network sends its queue size $q_j(t)$ to all other computers in the network. A node i receive this information from node j delayed by a finite amount of time τ_{ij}.

Each node then uses this information to compute its estimate of the network average of the number of tasks in all n queues of the network. Based on the most recent observations, the local estimate of the network average is computed by the ith node as Node i then compare its queue size $q_j(t)$ with its estimate of the network average by estimating its excess load $q_i(t) - (\sum_{j=1}^{n} q_j(t-\tau_{ij}))$. If it's excess load is greater than zero or some positive threshold, the node sends some of its tasks to the other nodes. If it is less than zero, no tasks are sent. Further, the tasks sent by node i are received by node j with a delay h_{ij}. The load balancing algorithm decides how often to do load balancing and how many tasks are to be sent to each node.

The mathematical model of the task load dynamics at a given computing node is given by

$$\frac{d}{dt}x_i(t) = \lambda_i - \mu_i + u_i(t) - \sum_{j=1}^{n} p_{ij}\frac{t_{pi}}{t_{pj}}u_j(t-h_{ij}) \quad (11)$$

Where $x_i(t)$ is the expected waiting time

$\lambda_i \geq 0$ is the rate of generation of waiting times on the ith node caused by the addition of tasks.

$\mu_i \geq 0$ is the rate of reduction in waiting time caused by the service of tasks at the ith node

$u_i(t)$ is the rate of removal (transfer) of the tasks from node at time by the load balancing algorithm at node.

p_{ij} is the fraction of the ith node's tasks to be sent out that it sends to the ith node.

The quantity $-p_{ij}u_j(t-h_{ij})$ is the rate of increase (rate of transfer) of the expected waiting Node j performs this computation for all the other nodes and then portions out its tasks among the other nodes according to the amounts they are below the local average, that is

$$p_{ij} \triangleq \frac{sat(x_j^{avg} - x_i(t-\tau_{ji}))}{\sum_{i\ni i!=j} sat(x_j^{avg} - x_{i(t-\tau_{ji})})} \quad (12)$$

The model was shown to be consistent in that the total number of tasks is conserved and the queues were always nonnegative and also the system was shown to be always stable.The comparative study of the papers is tubulised as shown in Table I

D. Proposed Work

The Architecture of the proposed system will be consisting of the server and the number of clients where

DOI: 03.LSCS.2013.1.37

TABLE I. COMPARISON OF ABOVE LOAD BALANCING TECHNIQUES

Papers parameters	"On Delay Adjustment for Dynamic Load Balancing in Distributed Virtual Environments."	"Communication-Aware Load Balancing for Parallel Applications on Clusters"	"The Effect of Time Delays on the Stability of Load Balancing Algorithms for Parallel Computations"
Environment	Distributed virtual environment.	Clusters computation	Cluster Computation
Technique	The change in the load of the servers due to network delay would affect the performance of the load balancing algorithm.	• This scheme is designed to reduce the performance gap between the effective speed of CPU and network resources. • Behavioural model is introduced to capture the resource requirements. COM aware uses this model for the effective BW utilization	The effects of delays in the exchange of information among CEs, and effects impose on the design of a load balancing strategy.
Points to be focussed	• Load Dynamics is considered • Whether existing load or the newly added load to be transferred is not considered	• Load Dynamics is considered • Selection process of the node to transfer the load is to be considered	• Load Dynamics is not considered • Selection process of the node to transfer the load is to be considered

clients run with the interactive applications whose load will be continuously varying. At the server side highest priority will be given to the interactive applications.

The proposed simple design model is as shown below Fig. 1

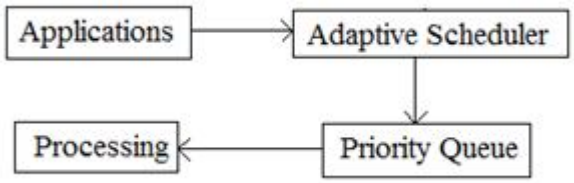

Fig. 1. Proposed Design

CONCLUSIONS

Due to communication delays among servers, the load balancing process may be using outdated load information to conduct load migration. This would significantly affect the performance of the load balancing algorithm. We need to reduce the latency for Interactive applications such that it is in lined with the degree of load change. From literature survey study we find that latency for a network has high impact of software overhead then delay of a network. Hence we have proposed the algorithm in which the queuing delay is minimized by giving the highest priority to the Interactive applications.

REFERENCES

[1] Yunhua Deng and Rynson W.H. Lau. "On Delay Adjustment for Dynamic Load Balancing in Distributed Virtual Environments", IEEE Transactions On Visualization And Computer Graphics, Vol. 18, No. 4, April 2012.

[2] Xiao Qin, Hong Jiang, Adam Manzanares, Xiaojun Ruan and Shu Yin, IEEE "Communication-Aware Load Balancing for Parallel Applications on Clusters" IEEE Transactions OnComputers, Vol. 59, No. 1, January 2010.

[3] John Chiasson, Zhong Tang, Jean Ghanem, Chaouki T. Abdallah, J. Douglas Birdwell, Majeed M. Hayat, and Henry Jérez, "The Effect of Time Delays on the Stability of Load Balancing Algorithms for Parallel Computations", IEEE Transactions On Control Systems Technology, Vol. 13, No. 6, November 2005

[4] Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing of SAMR applications on distributed systems. In Proc. ACM/ IEEE Conference on Supercomputing, pages 24–24, 2001 Dynamic Load Balancing and Pricing in Grid Computing with Communication Delay, Journal Of Grid Computing, Volume 6, Number 3 (2008)

[5] M. Harchol-Balter and A.B. Downey, "Exploiting Process Lifetime Distributions for Dynamic Load Balancing," ACM Trans. Computer Systems, vol. 15, no. 3, pp. 253-285, 1997.

[6] X.-D. Zhang, L. Xiao and Y.-X. Qu, "Improving Distributed Workload Performance by Sharing Both CPU and Memory - Resources," Proc. 20th Int'l Conf. Distributed Computing Systems (ICDCS '00), pp. 233-241, 2000.

[7] X. Qin, "Design and Analysis of a Load Balancing Strategy in Data Grids," Future Generation Computer Systems, vol. 23, no. 1, pp. 132 - 137, 2007.

[8] J. Cruz and K. Park, "Towards Communication-Sensitive Load Balancing," Proc. 21st Int'l Conf. Distributed Computing Systems, pp. 731-734, Apr. 2001.

[9] M. Hayat, S. Dhakal, C. Abdallah, J. Douglas Birdwell, and J. Chiasson. "Dynamic time delay models for load balancing. Part II: A stochastic analysis of the effect of delay uncertainty". In Proc. CNRS-NSFWorkshop: Advances in Control of Time-Delay System, 2003.

[10] Abbas Karimi, Faraneh Zarafshan, Adznan b. Jantan, A.R. Ramli1, M. Iqbal b.Saripan. "A New Fuzzy Approach for Dynamic Load Balancing Algorithm.". (IJCSIS) International Journal of Computer Science and Information Security Vol. 6, No. 1, 2.

DOI: 03.LSCS.2013.1.37